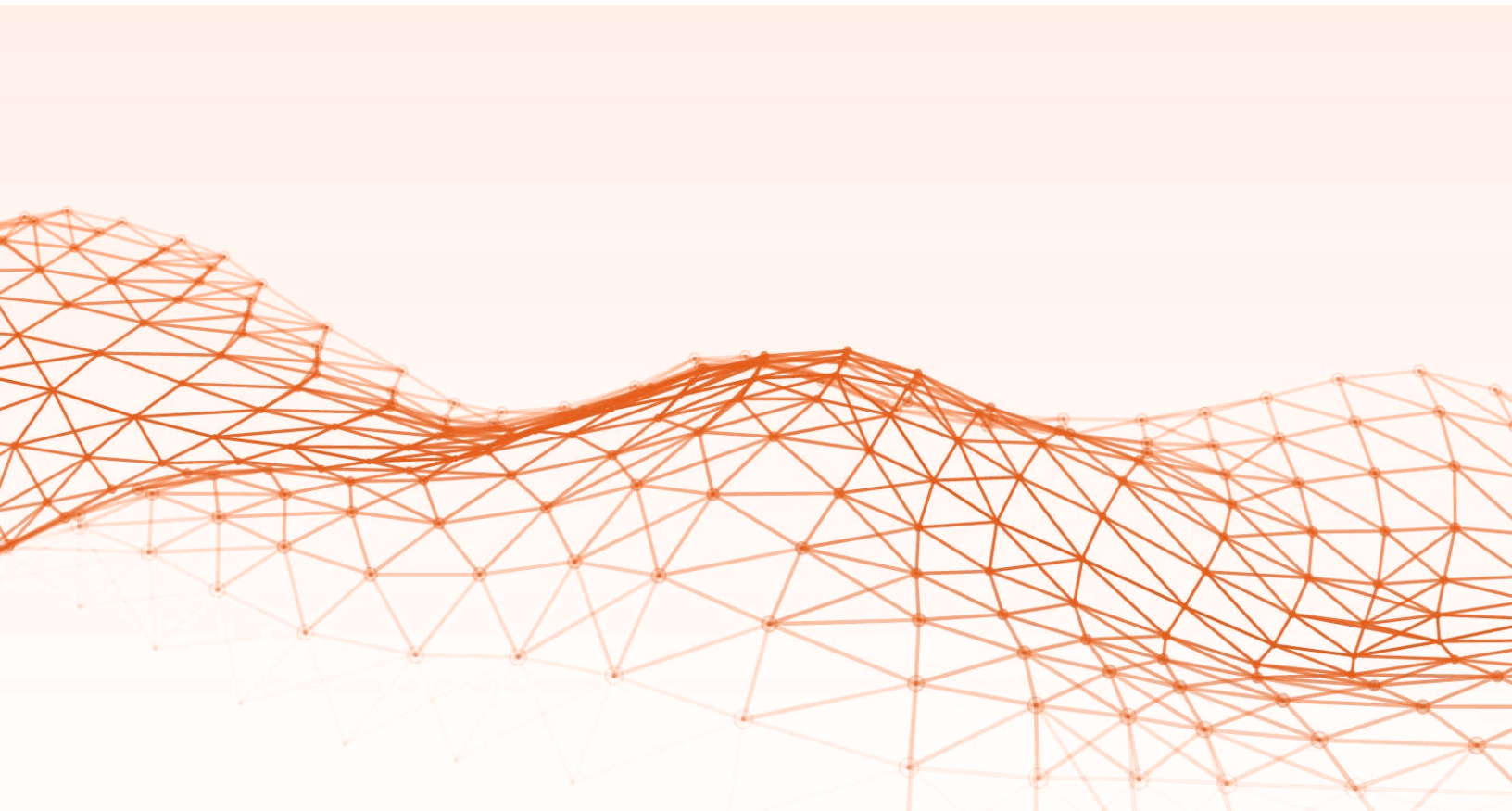
tetrate

Service Mesh Handbook

Everything you need to know about service meshes including Istio, mTLS, zero trust, best practices for adoption and more



About Tetrate

Rooted in open source, Tetrate was founded to solve the application networking and security challenges created by modern computing so enterprises can innovate with speed and safety in hybrid and multi-cloud environments. As applications evolve into collections of decentralized microservices, monitoring and managing the network communications and security among those myriad services becomes challenging. This is why some of the largest financial institutions, governments and other enterprises rely on Tetrate to deliver modern application networking and security on a foundation of Zero Trust. http://www.tetrate.io.

Table of Contents

Executive Summary

In the modern era, software systems are deployed in the cloud, on multiple clouds, private data centers and even edge locations. The infrastructure is dynamic. The software comprises hundreds and thousands of microservices that may be implemented in multiple programming languages. The infrastructure and application development follow DevOps practices for continuous delivery. Security is integrated into the process following DevSecOps practices. Different components of the system are released constantly.

This was a boon for productivity and flexibility, but brought on new problems of management, control and policy enforcement. All these microservices implemented in multiple languages somehow need to interact. Developers and administrators need to understand the flow of information, be able to detect and mitigate problems and secure the data and the infrastructure.

Enter the service mesh - built on open source Istio. An Istio-powered service mesh provides networking, security and observability for microservices running in Kubernetes and other container orchestration environments, as well as aligns various teams in complex distributed environments. An enterprise service mesh will help organizations manage their microservices-based applications and usher legacy apps into the cloud, making it a critical technology in modern, cloud-native and microservices architectures. The service mesh :

- **Offload Networking Concerns:** The service mesh externalizes all the networking concerns from the applications. Now they can be managed and updated centrally. By offloading all networking concerns to the service mesh, service developers can focus their efforts solely on their application and business logic.
- **Transparent Library Upgrades:** With a service mesh, you can upgrade your service mesh and everyone immediately enjoys the latest and greatest transparently. Traditionally, to introduce a change or upgrade to a client library, you would need to negotiate with each team individually, supporting multiple versions of libraries and across multiple programming languages.
- **Simpler Management of Cross-Cutting Concerns:** As a central component that touches all of your services, the service mesh can handle cross-cutting concerns - such as observability, health checks and access policy enforcement - across all services in your Kubernetes-based system.
- **Observability:** Service meshes provide extensive observability features. You can monitor traffic, collect metrics and gain insights into the behavior of your microservices. This visibility simplifies troubleshooting and aids in the early detection of security anomalies.
- **Policy Enforcement:** With a service mesh, you can define and enforce access control policies consistently across your microservices. This means you can easily implement security policies without modifying application code.

- **Multi-Cloud Support:** In a multi-cloud environment, service meshes abstract the underlying cloud provider differences, enabling applications to run across multiple cloud platforms with minimal code changes.

- **Zero Trust Security Out of the Box:** The service mesh can add a layer of security to an enterprise's inter-service communication by employing a zero trust approach to access and using mTLS to encrypt traffic for secure communication. Additionally, limiting access from application to application helps to ensure that a malicious attacker who exploits one service cannot move laterally through your network to exploit other services.

The service mesh also aligns developers, security, networking and platform teams by providing a common infrastructure layer that simplifies collaboration and ensures that all teams are on the same page regarding security policies and network configurations. By aligning developers, security, networking and platform teams, the service mesh streamlines operations and enhances the overall security and reliability of your cloud-native applications. Here's how:

- **Developers:** With the service mesh handling communication, security and reliability concerns, developers can concentrate on writing the core business logic of their applications and delivering business value. This results in more efficient development processes, improved application quality and faster time-to-market.

- **Security Teams:** Security teams benefit from the fine-grained access control and encryption provided by the service mesh. They can define and enforce security policies that are consistently applied, reducing risks.

- **Networking Teams:** Networking teams appreciate the simplified traffic management and observability features of the service mesh. It streamlines their tasks and enhances the reliability of the network.

- **Platform Teams:** For platform and DevOps teams, the service mesh simplifies the deployment of applications and their infrastructure. It aligns with CI/CD pipelines, enabling programmable and automated deployments while managing networking and security policies as code.

The service mesh provides a powerful platform for building and operating complex, distributed applications efficiently and securely across multi-cloud environments with less manual toil. The service mesh aligns developers, security, networking and platform teams by providing a common infrastructure layer that simplifies collaboration and ensures that all the teams are on the same page regarding security policies and network configurations. By aligning developers, security, networking and platform teams, the service mesh streamlines operations and enhances the overall security and reliability of your cloud-native applications. These benefits, especially developer productivity, are inordinately impactful given today's reliance on digital technologies. Increasing developer productivity, removing complexity and reducing toil provide a faster path to production and reduce time to market, which reduces time to value.

01 The Modern, Multi-Cloud Era

Rapid change is a constant in technology, but you might be forgiven for thinking that, at the moment, change is even more rapid than normal. There is a fairly predictable cycle of creative destruction in enterprise-scale information technology, but the last few years have been a bit different.

Since at least the microprocessor revolution, which began in the 1970s, there has tended to be a single technology shift in progress, at any given time, that has an outsized influence on technology change. In the 1970s and '80s, that change driver was the phenomenal acceleration of microprocessor power (which continues). In the 1990s and 2000s, it was the accelerating global adoption of the Internet and the World Wide Web, and the mass scramble of businesses to reach "Internet scale."

Our current moment is a bit different, though. We are in the middle of a large shift in how most things get done, powered by four transformations:

- Digital transformation, which was only accelerated by the global pandemic when it arrived in 2020
- Shifting infrastructure, from servers in organizationally-owned data centers to public cloud providers
- Shifting architecture, from monoliths to microservices
- Shifting processes, from long waterfall development and delivery cycles to agile approaches

Below, we'll describe how strong the forces of change are and how these shifts are affecting an organization's IT strategy and infrastructure. We will then introduce service mesh as a viable solution to many of these challenges.

Accelerating Cloud Adoption

In 2023, the global public cloud services market is expected to grow by approximately 21.7%, to total $597 billion. This is up from $491 billion in 2022. In the first quarter of 2023, enterprise spending on cloud infrastructure services reached approximately $64 billion according to IDC research. The allure of cloud computing for enterprises is well understood, after having been on the leading edge of the hype cycle for a decade and a half:

1. It's easy to provision infrastructure

2. Managed cloud services make huge swaths of technical expertise, effort, and organizational pain someone else's problem

3. The cloud enables a consumption-driven pricing model, reducing the need for up-front planning, budgeting, and spending

4. New and advanced technologies can be tried and adopted much faster, and at a much lower cost of entry, enhancing agility

Of course, no cloud is all silver lining. There are downsides to significant cloud adoption, many of which only become apparent a fair way into the journey. Chief among the gripes large cloud adopters have is the cost. The flip side of “pay as you go” is, “as you go, you pay.”

While it may be easier to have other people provision infrastructure and manage it for you, it's not cheap. It's also harder to see the big picture when applications are spread across many different environments. This can make troubleshooting more difficult. In addition, it’s harder to implement, enforce, and audit consistent security policies across a combination of clouds and company-owned data centers.

At the same time that cloud adoption accelerates, the data center is not going away anytime soon. While organizations race to the cloud, they'll be managing their existing application stock alongside their shiny new cloud applications for many years to come.

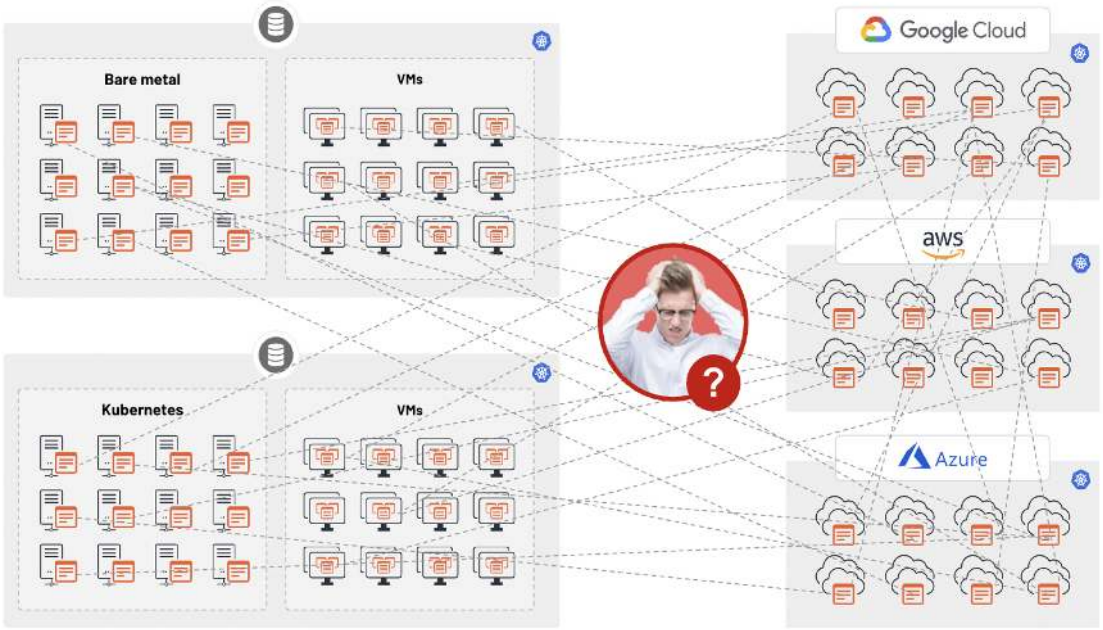
Modernization Momentum Continues

A twin transformation—so closely associated with cloud adoption that they are usually spoken of in the same breath—is application modernization. The main driver for decomposing monoliths to microservices is agility, a theme we'll see crop up a lot: ship more value faster with less overhead, use the right stack for the job, and align with DevOps processes and best practices. And, if you're moving into the cloud, it makes sense to be cloud native, which requires that you modernize your applications.

As always, there are trade-offs—or, at least, new point-in-time challenges to address. Where monolithic applications have the luxury of doing much of their work in a single block of machine memory (or at least in memory and on disk on a single machine), microservices must have a reliable network between them to do pretty much anything. Testing, dependency management, and troubleshooting are inherently more difficult in microservices applications because there are many more moving parts. For the same reason, microservices offer a broader attack surface with more potential attack vectors.

Security and Networking (Still) Impede Agility

Cloud-native applications have the benefit of declarative policy starting in the development process and extending into production. Increased automation promises more unified policy enforcement. But, mid-transformation, many organizations are still trying to connect and secure cloud-native applications with a grab bag of opaque, often manual, L3/L4 tooling like firewalls and VPNs. It remains slow and difficult to make changes to network and security policy; and, at the heart of things, it's difficult to make changes to institutional behavior. This results in more complexity and a lack of operational agility.



Precisely because many large organizations are only part of the way through several large, simultaneous technology transformations, many of the promised gains are yet to be realized, while relatively short-term, point-in-time challenges are yet to be overcome. The current state is in many ways an unfortunate worst-of-all-worlds scenario, with new complexity overwhelming the agility gains of moving to the cloud and microservices.

Unrealized Goals

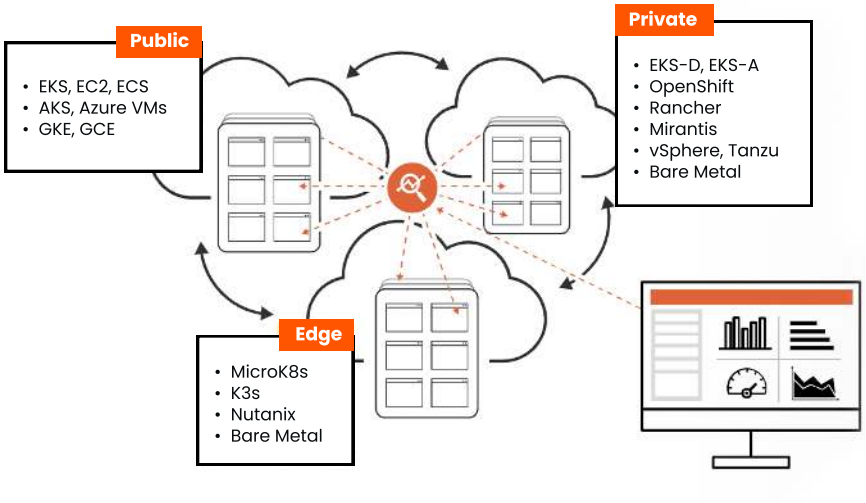
Agility	Security and Isolation	Business Continuity
• Traffic shaping and canary controls • Service discovery across multiple clusters and data centers • On-demand workflows for faster, safer rollouts • Faster application troubleshooting	• Multitenancy • Cross-cluster and cross-cloud security policies and access control • Application-level segmentation: fine-grained ingress and egress controls • Secure by default with removal of implicit trust between services	• High availability and resiliency for active-active deployments • Unified telemetry and service-level objective management

Mid-transformation, we see many organizations still struggling to achieve goals across three broad categories: agility, security, and business continuity. Agility is hampered by lack of visibility and control over what has become something of a sprawling mess spread among heterogeneous environments across data centers and different clouds, each requiring a different set of tools and practices. Security policy is harder than ever to enforce consistently across these environments, just as it's more important than ever to get right. And, even though the cloud offers near-instant scale, business continuity is at risk from the sheer difficulty of getting quality intel across a sprawling fleet, to allow operators to identify failures, and to route around them quickly enough to avoid outages.

Cloud != Cloud-Native

Just like moving to Hollywood doesn't make you a movie star, moving to the cloud doesn't make you cloud-native. You can't be cloud-native at scale without a modern, application-aware network.

Service Mesh: Agent of Cross-Cutting Change



As a dedicated infrastructure layer, Tetrate's service mesh is a vital component in modern, cloud-native architectures.

These are the very challenges that service mesh—and Tetrate as a company—were created to meet. Service mesh as a solution extracts the cross-cutting concerns of security, connectivity, observability, and reliability from application code and libraries and into a dedicated infrastructure layer. This layer can be managed centrally and implemented and enforced consistently across different environments, compute architectures, and language ecosystems.

In the following pages, we'll cover :

1. Service mesh fundamentals
2. Why you need a service mesh for Kubernetes
3. An overview of the business needs driving service mesh adoption
4. The service mesh ecosystem

02 What is a Service Mesh?

From the beginning, Google faced a challenge: processing massive, and fast-growing, amounts of data in a distributed fashion to provide needed services and power analytics. To meet this challenge, they transformed their internal application architecture from a centralized API gateway to decentralized microservices on a flat network. The service mesh approach evolved to help address concerns that arose in this transformation. And, now that many companies face similar challenges to Google's, the need for service mesh is universal.

The centralized gateway presented multiple challenges: from locality to cache coherency and increased overhead from extra hops. Two of the top challenges were cost accounting and shared-fate outages. It was impossible for the team running the API gateway to perform cost attribution to individual teams using the gateway—making it challenging to control costs and to do capacity planning—as well as being a common root cause for outages affecting business continuity.

This poor fit between a central API gateway and a decentralized service architecture prompted the development of a proto-service mesh, where communication from users to services, as well as communication from services to other services, was handled by sidecar proxies talking directly to each other instead of through a central API gateway.

A sidecar proxy is a piece of software that is put alongside each of the individual services that make up a microservices application. Applications depend on the sidecar proxies to handle communication tasks between services. The mesh is composed of sidecar proxies, with each instance of the proxy deployed beside a single service instance, as a sidecar.

The benefits of implementing service mesh at Google were striking. Teams were now able to easily do cost accounting, and shared-fate outages due to an individual team's misconfigurations were essentially eliminated. As the teams, and the organization, grew more confident with the mesh, they started to see other benefits they didn't originally expect: it was an incredible tool for implementing cross-cutting features for the entire platform.

For example, before the service mesh, an effort to add three identity access management (IAM) methods to all GCP APIs took 40 teams, each contributing around two engineers, more than six months to implement. It was an organizationally scarring experience. After the mesh was in place, a similar global API update to add policy checks to every resource took only two engineers a single quarter to implement.

Shortly after the service mesh experience at Google, the team started Istio to bring service mesh to the world. Listening to Istio's earliest users revealed a common set of challenges that the mesh uniquely addresses: not just shared-fate outages or cost attribution, but difficulty enabling developers to build features for users more quickly while the organization grapples with the transition imposed by modernization and move-to-cloud.

Features, Capabilities, and Architecture

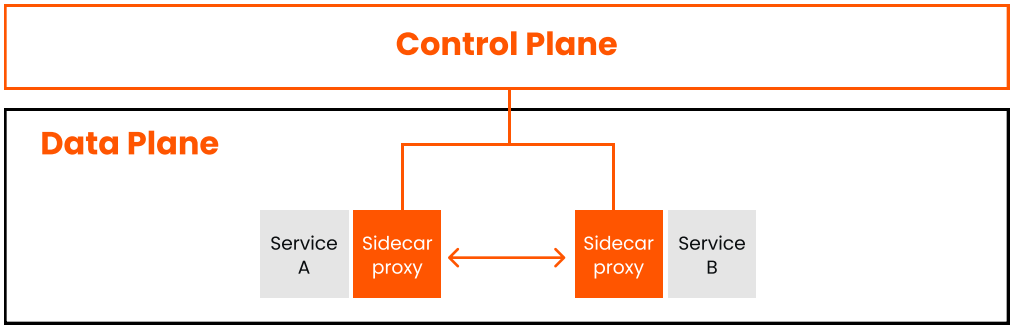


Figure 2.1: service mesh architecture[1]

The service mesh provides features that are critical for running modern server-side software in a way that's uniform across your stack and decoupled from application logic. It is an infrastructure layer that sits between application components and the network via a proxy. These app components are often microservices, but any workload from serverless containers to traditional n-tier applications in VMs or on bare metal can participate in a mesh. Rather than each component communicating directly with other components over the network, the proxies mediate that communication. These proxies form the data plane, providing a ton of capabilities for implementing security and traffic policy, as well as producing telemetry about the services they're deployed with. They're configured by a control plane dynamically at runtime so policy can be updated on-the-fly (Figure 2.1).

Common capabilities offered to applications by the mesh include:

- Service discovery
- Resiliency: retries, outlier detection, circuit breaking, timeouts, etc.
- Client-side load balancing
- Fine-grained traffic control at L7 (not L4!): route by headers, destination or source, etc.
- Security policy on each request, rather than once per connection
- Authentication, rate limiting, arbitrary policy based on L7 metadata
- Strong (L7) workload identity
- Service-to-service authorization
- Metrics, logs, and tracing

All of these capabilities are available to some degree or other in frameworks and libraries today, for example, Hystrix, and Spring Boot and Spring Cloud. (Most innovations in the cloud-native architecture space have focused on the Java virtual machine.) However, by pulling these capabilities

[1] Istio recently announced "Ambient Mesh" — an experimental, "sidecar-less" deployment model for Istio. While there are specific use cases where this approach will yield benefits, Ambient Mesh has yet to gain "production-ready" status in enterprises. Learn more about Ambient Mesh on Tetrate's blog post.

out of the application components and into a common infrastructure layer, we get centralized control with distributed enforcement that's just not possible with the rogues' gallery of frameworks and libraries littering the landscape.

This common infrastructure layer gives organizations with many applications, written in many languages, and deployed across many different environments the benefits of:

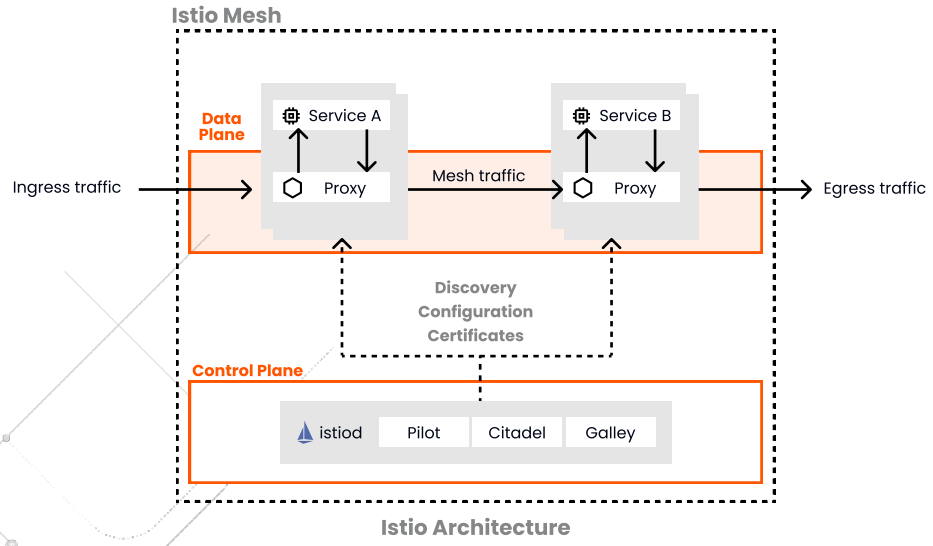
- Centralized visibility and control
- Consistency across the entire fleet
- Ease of change through policy and configuration implemented as code
- A lifecycle for these capabilities separate from any application lifecycle - which means you don't have to upgrade and redeploy an application to apply, for example, mTLS between components

All of this leads to greater developer efficiency, since these considerations are simply removed from the requirements an application must meet. Any concerns or problems in these areas are no longer a concern of the developer.

Istio: the Industry Leading Mesh Implementation

Istio is an open-source service mesh that helps organizations run distributed, microservices-based applications. It provides a way to secure, connect and monitor microservices. Istio layers transparently onto existing distributed applications and sits as an overlay between the layers of distributed applications. Istio works natively with Kubernetes (K8s) only, but its open source nature makes it possible for anyone to write extensions enabling Istio to run on any cluster software. Istio includes APIs that let Istio integrate into any logging platform, telemetry or policy system. Istio enables intelligent application-aware load balancing from the application layer to other mesh-enabled services in the cluster. It bypasses the rudimentary kube-proxy load balancing.

Istio's core consists of a control plane and a data plane, with Envoy as the default data-plane agent. The following diagram shows the various components of each plane:



Istio Architecture

Istio's powerful control plane brings vital features including:

- Secure service-to-service communication in a cluster with TLS encryption, strong identity-based authentication and authorization
- Automatic load balancing for HTTP, gRPC, WebSocket and TCP traffic
- Fine-grained control of traffic behavior with rich routing rules, retries, failovers, and fault injection
- A pluggable policy layer and configuration API supporting access controls, rate limits and quotas
- Automatic metrics, logs and traces for all traffic within a cluster, including cluster ingress and egress

Istio is designed for extensibility and can handle a diverse range of deployment needs. Istio's control plane runs on Kubernetes and you can add applications deployed in that cluster to your mesh, extend the mesh to other clusters, or even connect VMs or other endpoints running outside of Kubernetes.
As organizations accelerate their moves to the cloud, they are, by necessity, modernizing their applications as well. But shifting from monolithic legacy apps to cloud-native ones can raise challenges for DevOps teams. Developers must learn to assemble apps using loosely coupled microservices to ensure portability in the cloud. At the same time, ops teams must manage the new cloud-native apps within increasingly large hybrid and multi-cloud environments. Istio enables developers and DevOps teams to secure, connect and monitor microservices, so they can modernize their enterprise apps more swiftly and securely.

> "A service mesh is the right architecture for the enforcement of authorization policies since the components involved are moved out of the application and executed in a space where they can form a security kernel that can be vetted

National Institute of Standards and Technology (NIST) SP 800-204B

The mesh is the security kernel for microservices-based applications. As a result, the choice of service mesh implementation techniques has a direct impact on application and information security. The level of intensity around development, bug fixes, and security patches should match the level of trust you expect to place in the mesh as your application security kernel.

For example, Istio is the reference implementation for microservices security standards developed by the US National Institute of Standards and Technology (NIST), with increasing use in the Department of Defense and the federal government. The NIST microservices security standards are found in the NIST Special Publication 800-204 series (SP 800-204, SP 800-204A, SP 800-204B, and SP 800-204C). Taking microservices security standards further, Istio is also NIST's reference implementation for comprehensive zero trust principles in NIST Special publication 800-207A. The NIST Zero Trust Architecture Model for Access Control in Cloud-Native Applications in Multi-Cloud Environments. Tetrate has been a major contributor to these standards.

Istio is the most widely used (CNCF, 2022) and most robustly supported service mesh, with a history of prompt CVE patches, paid security audits, and currently active bug bounties. And Istio is the only service mesh with an ecosystem that enjoys both grass-roots support and support from multiple institutions, large and small.

Istio is also evolving in tandem with the Kubernetes ecosystem to offer an ever more seamless experience. The new 2.x Kubernetes networking API now uses the Istio networking API. And, as Kubernetes projects like Knative expand the Kubernetes ecosystem, standardization around Istio promises to make service mesh practices seamless across environments and deployment patterns.

The Power of Service Mesh and Kubernetes

The synergy between Kubernetes and service meshes is powerful as the service mesh builds on top of the basic Kubernetes networking model. Kubernetes offers a flexible and extensible framework, making integration with service meshes seamless. While Kubernetes is a widely deployed platform for large-scale distributed systems, out of the box it doesn't address all the needs of complex enterprise systems deployed across multiple clouds and private data centers. This is where an Istio-based service mesh fills that gap, providing vital connectivity, management and observability across Kubernetes clusters while seamlessly connecting multiple clusters across diverse cloud infrastructures. It reduces the operational burden on devops teams and makes it easier to implement advanced deployment strategies, ensuring that your application performs at a high level as it grows and evolves.

Here is why you need a service mesh with Kubernetes:

- **Scalability:** Kubernetes is designed for application lifecycle management through declarative configuration, while a service mesh provides inter-application traffic, security management and observability. As your application grows and the number of services and endpoints increases, a service mesh can handle the complexity more effectively and in a scalable manner.
- **Traffic Control:** A service mesh decouples traffic management from Kubernetes, allowing for more elastic and granular control over traffic routing, load balancing and API calls between services and endpoints. This is achieved through sidecar proxies that handle the traffic at the application layer, providing more flexibility and control than the global kube-proxy settings in Kubernetes.
- **Observability:** Service mesh in Kubernetes offers logging, tracing and monitoring capabilities, making it easier to identify and troubleshoot issues such as service latency problems. Integrating with tools like Prometheus and Jaeger, a service mesh provides visibility into the behavior of the entire network of distributed microservices.
- **Security:** A service mesh enables secure communication between services through encryption and authentication. It allows you to enforce policies to allow or deny connections, ensuring that only authorized services can access your production environment.

- **Simplifying Development:** By offloading the responsibility of managing network traffic, security and observability to the service mesh, developers can focus more on writing the core business logic of the application, making microservices simpler and more lightweight.
- **Advanced Deployment Strategies:** Service mesh in Kubernetes enables canaries, rolling upgrades, blue/green and other advanced deployment strategies by providing intelligent routing and control over API calls and traffic flow between endpoints and services

For large systems - in particular, systems composed of multiple Kubernetes clusters - the service mesh becomes a standard add-on. Once enterprises and federal agencies begin working with multiple clusters, which might spread across different clouds, the service mesh becomes essential. Service meshes are designed to work seamlessly with various Kubernetes distributions and container orchestration platforms including AKS, EKS, GKE and OKE ensuring that you can adopt and manage multi-cloud environments without being locked into a specific vendor. Furthermore, you can define and enforce consistent network policies, security policies and access control policies using a service mesh. This simplifies governance and compliance efforts in a multi-cloud setup, helps eliminate inefficiencies and reduces friction between networking, security and development teams.

Align Developer, Security, Networking and Platform Teams

The service mesh aligns developers, security, networking and platform teams by providing a common infrastructure layer that simplifies collaboration and ensures that all teams are on the same page regarding security policies and network configurations.

Without a service mesh platform, contemporary applications with a microservices-based architecture would have a much larger overhead in terms of design, development, and maintenance. Right from maintaining separate business logic and configuration specs to complex authentication and authorization implementations that are custom to the application, developers will have to spend a lot of time gluing together disparate technology components.

With an application developed through a service mesh implementation, developers can let the platform do much of the heavy lifting in terms of inter/intra communication, traffic routing between microservices, load balancing, policy enforcement, workflow and configuration safeguards. This allows development teams to focus primarily on using the right design patterns, efficient business logic, and other aspects. By aligning developers, security, networking and platform teams, the service mesh streamlines operations and enhances the overall security and reliability of your cloud-native applications. Here's how:

- **Developers:** With the service mesh handling communication, security and reliability concerns, developers can concentrate on writing the core business logic of their applications and delivering business value. This results in more efficient development processes, improved application quality and faster time-to-market.
- **Security Teams:** Security teams benefit from the fine-grained access control and encryption provided by the service mesh. They can define and enforce security policies that are consistently applied, reducing risks.

- **Networking Teams:** Networking teams appreciate the simplified traffic management and observability features of the service mesh. It streamlines their tasks and enhances the reliability of the network.

- **Platform Teams:** For platform and DevOps teams, the service mesh simplifies the deployment of applications and their infrastructure. It aligns with CI/CD pipelines, enabling programmable and automated deployments while managing networking and security policies as code.

The service mesh provides a powerful platform for building and operating complex, distributed applications efficiently and securely across multi-cloud environments with less specialized expertise in each cloud and less manual toil. Developers, security, networking and platform engineers are able to perform their jobs better individually while collectively innovating faster. These benefits, especially developer productivity, are inordinately impactful given today's reliance on digital technologies. Increasing developer productivity, removing complexity and reducing toil provide a faster path to production and reduce time to market, which reduces time to value.

03 Business Drivers: Security, Agility and Business Continuity

Although we're in the midst of a large-scale technological transformation, it's the latest in a long history of technological transformations. The curious mind may wonder why we seem to rebuild our software every 10 years. One answer to that question is the cyclical nature of creative destruction: we must take things apart to innovate on particular pieces of technology before knitting them back together into a coherent whole.

Twenty years ago, that creative dismantling was the break-out of business applications from mainframes to rebuild them with n-tier architectures on commodity hardware sliced up into virtual machines (VMs). Over the last decade, we've begun to decompose those monoliths into microservices - which drive, and are also driven by, process innovations like agile and DevOps/ DevSecOps. We have also stuffed those bits and pieces into orchestrated containers. Simultaneously, we're breaking infrastructure free from static servers and network devices in data center racks and floating them up to on-demand, automated computing and networking infrastructure in the cloud.

This ontological transformation is not cost-free, unfortunately. Organizations in the thick of this cycle of creative destruction suffer transformation fatigue, as the promised agility and efficiency gains are slow to materialize for many. The expected gains remain elusive largely because the creative destruction cycle is incomplete: we must still re-weave the newly innovated pieces—microservices, DevOps, containers, and cloud—back together into a coherent whole that is better, stronger, faster than before. Across the domains of application security, connectivity, observability, and reliability, the service mesh is the warp and weft of that whole, interconnected cloth.

On this final leg of our transformation journey, the weaving together of applications with a service mesh yields three main business benefits: security, agility, and business continuity.

Security and Zero Trust Architecture

Zero Trust architecture (ZTA) is a crucial security framework that addresses the challenges posed by the evolving threat landscape, remote work, cloud adoption and compliance requirements. Achieving a ZTA is a key goal for many enterprises and government agencies today that are trying to reduce risk in their environments. Infrastructure and operations networking leaders and their teams are increasingly asked to apply zero trust concepts to their infrastructure. In fact, the Executive Order 14028 (EO 14028) on Improving the Nation's Cybersecurity pushes agencies to adopt zero trust cybersecurity principles and adjust their network accordingly by 2024. Zero trust replaces implicit trust with continuously assessed risk and trust levels, based on identity and context. By adopting ZTA, organizations can enhance their security posture, reduce risk and better protect their valuable assets and data.

Unfortunately, some believe that ZTA can be reduced to simply using encryption for all data in transit. While that's part of it, the scope of ZTA is bigger than you may think. To get a sense of what it would mean to fully implement a zero trust architecture, a useful rubrik is: if you took all of your application components and exposed them to the open Internet, what would you need to change? You're at zero trust when the answer to that question is, "nothing bad."

The term zero trust can be confusing. The full description would be “zero implicit trust” or “zero assumption of trust.” In a mesh, each service’s proxy authorizes and authenticates each message, re-confirming trust for each and every message between services. And yes, the messages are encrypted.

Adopting a full zero trust posture for most organizations won't happen overnight. But adopting a service mesh offers a standard, consistent way to close the gap across much of your application fleet.

Zero Trust Security Out of the Box

Service mesh provides a comprehensive set of out-of-the-box security features available to protect every application component in your fleet, including:

- Strong workload identity
- End-to-end mTLS
- Authentication
- Dynamic, fine-grained authorization
- Envoy as a Policy Enforcement Point (PEP) protecting every workload, whether that's microservices in Kubernetes or VMs in a data center
- Centralized policy authorship with global/distributed policy enforcement

Deep Visibility

The Envoy data plane provides deep visibility into application activities, enabling a feedback loop to improve policy and support real-time detection of attacks, and threat hunting to minimize likelihood of cybersecurity incidents and reduce impact to the business.

Scaling Security Engineering

The global management plane coordinates control planes spanning heterogeneous infrastructure. It also serves as a common administration and collaboration point for security, networking, and application teams. This allows features and configurations to be prescribed, designed, implemented, and delivered within the continuous development process and tooling, offering force-multiplying scale to security engineering. Developers can do most work within guardrails established by security, bringing security and app dev into close collaboration, and bringing the speed and flexibility of agile processes to the security realm.

The control and data planes installed in the runtime environment offer consistent security features across heterogeneous app architectures and application stacks. The single admin/ops plane ensures consistent configurations across multiple infrastructure-as-a-service (IaaS) providers, such as AWS, Microsoft Azure, and Google Cloud Platform, and on-premises data centers.

Security Standards for Microservices

Tetrate collaborates with NIST to define and promote Zero Trust security standards. Tetrate founding engineer Zack Butcher is co-author of several NIST Special Publications including the 800-204 series (SP 800-204, SP 800-204A, SP 800-204B, and SP 800-204C) and SP 800-207A that sets the U.S. Government security standards for microservices applications. Those standards call out service mesh as the preferred technology for microservices security and Istio as the reference implementation.

Agility

In the world of dynamic, distributed, and heterogeneous services, the mesh offers centralized governance with local enforcement. By concentrating the cross-cutting application concerns of security, connectivity, observability, and reliability into a dedicated infrastructure layer shared by every application, the mesh makes everything more similar, yielding increased operational efficiency. The ad-hoc, semi-manual, semi-scripted bundles of baling wire and chewing gum surrounding and unique to each application's deployment and maintenance can be unwound and replaced with a standard set of automated processes common to every app.

Out-of-the-Box Features for Agility

The service mesh helps us realize agility in our organization by:

- Centralizing cross-cutting concerns (like encryption in transit) into the platform layer, so application developers don't need to worry about them
- Giving us confidence in application rollout and deployment through fine-grained traffic control enabling application canaries, progressive rollouts and A/B testing
- Failover and failback allow you to move quickly but safely

Business Continuity

As with agility and security, we see business continuity threatened by the increasing complexity and opacity of modern deployments, especially as organizations struggle to manage traditional workloads in tandem with their transformation efforts. Heterogeneity makes it hard to implement changes. Complexity-driven opacity makes it hard to even understand what's going on.

Service mesh can help solve this problem, driving business continuity in at least two critical ways. First, by offering seamless traffic shifting across failure domains. The mesh can be smart about locality when all systems are healthy and failover - shifting traffic away from unhealthy to healthy resources in other clusters, regions, or even clouds, all before downstream requests—and customers—are affected.

Service mesh helps operators implement "safety first" approaches such as canary rollouts (only sending new code to a small percentage of production traffic), A/B testing (assessing the impact of new code for some users against performance of the previous code base with other users), and failback as a way to return to a "known good" status.

Also, by providing consistent and global observability capabilities across applications, the mesh enables a coherent picture of what healthy applications look like, allowing app and ops teams to identify and rectify pathological behavior before it escalates. When failures do occur, the visibility and observability capabilities of the mesh allow for faster troubleshooting and lower mean time to resolution (MTTR) per incident.

Out-of-the-Box Features for Business Continuity

Service mesh provides a great set of features out-of-the-box to help ensure business continuity for applications deployed in the mesh, including:

- Application-level operational metrics (rate of traffic, rate of errors, latency) per request
- Distributed tracing and consistent access logging
- Resilience capabilities like timesouts, retries, outlier detection, and circuit breaking
- Service discovery and load balancing (including across clusters and clouds)

04 Service Mesh Ecosystem

As a dedicated infrastructure layer to handle cross-cutting security, connectivity, observability, and reliability concerns, the mesh sits on top of lower-level compute and networking infrastructure, mediating application-level communication between individual components as well as the outside world. As such, service mesh is part of a broader ecosystem and provides key integration points to critical systems outside its primary domain of responsibility.

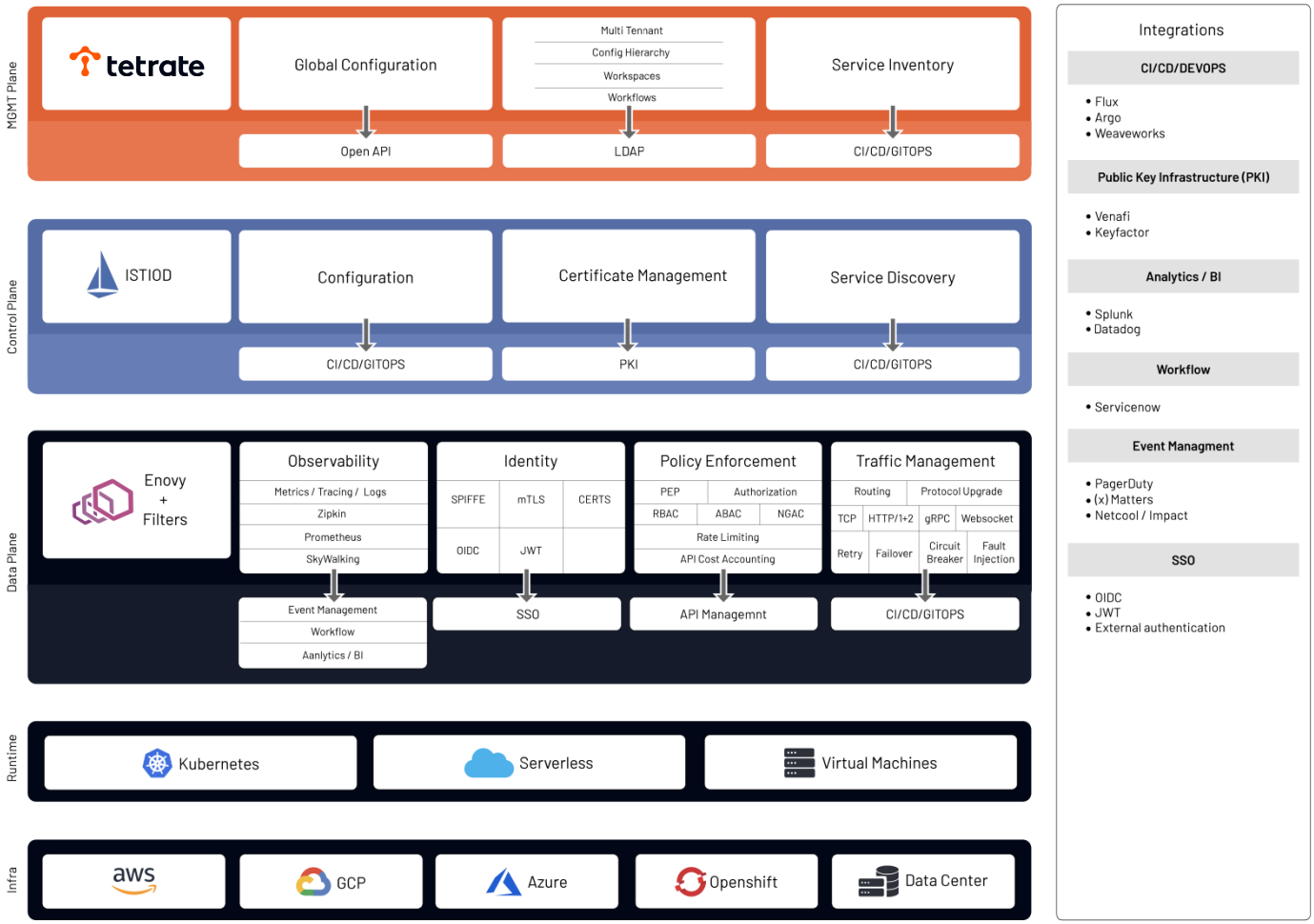


Figure 4.1: *Service mesh ecosystem and integrations*

Service Mesh Integrations

CI/CD/DevSecNetOps/GitOps: Because Istio is driven by policy-as-code, it fits naturally into existing git-driven workflows such as other Kubernetes objects. Integrations include:

- Argo
- Flux
- GitLab
- GitHub
- Jenkins
- Flagger

Public key infrastructure (PKI): Istio manages certificates as part of its built-in mTLS capabilities. Its internal certificate authority (CA) can be configured with a root certificate & signing certificate and key, DNS certificates, and custom CA integration using Kubernetes certificate signing requests. Tetrate Service Bridge (TSB) offers extended integration with third-party public key infrastructure providers including:

- Keyfactor
- Venafi
- Vault
- cert-manager
- Cloud Provider PKI (AWS Certificate Manager, Google Certificate Authority Service, Azure Key Vault)

Analytics/business intelligence (BI): Getting consistent and actionable runtime data out of your applications is a core value proposition of service mesh. Sharing that data with your analytics and business intelligence infrastructure is part of the fun. Integrations include:

- DataDog
- Splunk
- Elasticsearch

Business process/workflow: TSB allows you to plug into third-party workflow automation systems to, well, mesh with the way you already do business. Integrations include:

- ServiceNow
- C3 Ai

Event management: TSB integrates with your event management system. Think of it as an extension of observability capabilities of the mesh to fit into the way you handle event alerts and notifications: modern visibility and observability injected directly into your organization's brain. Integrations include:

- IBM Netcool/Impact
- Pagerduty
- xMatters

Service identity, authentication, and end-user credentials/SSO: Identity, authentication, and authorization are central concerns of the mesh. Out of the box, you get runtime service-to-service identity. For runtime integration with your existing identity provider, TSB offers OIDC and JWT support. TSB's management plane also syncs with your enterprise directory service to let you define and manage access policy, using the same users and roles that the rest of your organization does. Integrations include:

- OKTA
- Ping Identity
- Microsoft Azure AD
- IBM Security Verify
- Oracle IAM

05 Tetrate Service Bridge

Tetrate Service Bridge (TSB) is Tetrate's implementation of the powerful Istio open-source project, allowing you to manage, observe and secure your services without having to change your application code. TSB unburdens your operations and development teams by simplifying service delivery across the board, from traffic management and mesh telemetry to securing communications between services. TSB allows organizations to deploy their services to cloud providers that make the most sense for them based on cost, compliance or other strategic factors. Instead of being locked into a single vendor, organizations can use TSB to connect all of their clusters and services together regardless of the cloud that each runs on.

Tetrate Service Bridge provides a single pane of glass for observability, visualization, health checks, policy management and policy enforcement. It also collects telemetry data across heterogenous compute environments. Offloading these concerns - such as authentication, authorization, metrics collections, and health checks - to a central component is a game changer for DevOps, application developer productivity and multi-cloud adoption.

Edge-to-Workload Topology

TSB's topology spans 1) the application edge, with a Tier 1 gateway configuration; 2) application ingress, with ingress gateways in each cluster; and 3) a sidecar next to each service.

Application edge: Envoy-based, application-aware L7 gateway smart and flexible enough to facilitate modern multi-cluster deployments.

- Request-level traffic control across all compute clusters
- Traditional north-south API gateway functionality
- Single sign-on (SSO) for applications deployed in the mesh, without changes to the applications themselves
- Web Application Firewall (WAF) functionality for east-west in addition to north-south
-

Application ingress: Envoy-based, application-aware ingress load balancer combines service mesh ingress with backend API gateway functionality.

- Kubernetes ingress & service load balancing
- Traffic shifting between VMs and Kubernetes across cloud and on-premises
- Traditional east/west API gateway functionality, such as rate limiting and credential management
- Fault tolerance capabilities like timeouts, retries, and circuit breakers

Application egress: Envoy-based, application-aware egress load balancer gives you the capabilities to control how your applications communicate with services outside of the mesh.

- Hostname and SNI-based allow/deny listing
- Control which services can communicate out with fine-grained access control
- Traffic control for communicating with services outside of the mesh, including load balancing, timeouts, retries, and circuit breakers
- Credential exchange - store and maintain API keys or client certificates on the egress proxy rather than the application
- Detailed metrics to understand how you're consuming services outside of the mesh

Service mesh: In-cluster, Envoy sidecars connect individual workloads, serving as per-workload gateways. The Istio control plane applies configuration to sidecar proxies, as directed by the Tetrate's global management plane, and sends telemetry data back to it.

Global management plane: A central management plane coordinates policy, configuration, observability, and lifecycle across an entire application network topology to offer:

- Centralized management
- Multi-tenancy
- Global service inventory
- Configuration safeguards

Tetrate's Enterprise Service Mesh Features and Capabilities

Next-Generation API Governance

Because the mesh puts gateways around each workload and the management plane maintains a global view of application traffic, there is no practical difference between north-south and east-west traffic in a Tetrate-managed mesh. There is just application traffic. Out-of-the-box API management functionality may be applied at every layer: application edge, application ingress, and between services at the sidecar proxy. Those API-focused features include:

- **OpenAPI:** Configure gateways and the mesh with your OpenAPI spec
- **CORS:** Policy configuration
- **Authn/z:** mTLS, OIDC, JWT, IP blacklist/whitelist, integration with external auth services
- **Credential management**
- **Rate limiting:** Edge-to-workload and service-to-service
- **Fault tolerance:** Timeout, retry, circuit breaker
- **Transforms:** Custom header and body transforms for both request and response
- **Wasm:** Deploy custom Wasm filters to Envoy

Web Application Firewall (WAF)

TSB offers the inclusion of a WAF built on the Coraza project—implemented as an Envoy filter and configured by the global management plane—at every gateway and sidecar with support for reusable rulesets and the OWASP core rule set.

Global Application and Organizational Awareness

- **Centralized management:** While Istio is the de facto standard service mesh for single-cluster Kubernetes, TSB adds a global management plane that extends Istio to multi-cluster, multi-cloud, and hybrid cloud deployments on any compute.

- **Secure multi-tenancy, workspaces, and hierarchical permissions:** TSB makes it easy to control who in your organization can change what, audit those changes, and ensure your application deployment conforms to best practices for security. Tenant, workspace, and application constructs provide permissions guardrails between application teams operating in a shared cluster environment, so each team can have ownership over its own application resources without stepping on each other's toes. It's also possible to give infrastructure admins privileges specific to their area of expertise. For example, security admin permissions may be configured to allow updates to TLS certificates, but deny adjustments to traffic routing rules.

  Hierarchical permissions enable admins to configure top-level default policy that is automatically inherited down to tenants, workspace, and application levels, ensuring consistent policy enforcement for every application. This allows security, network, and application administrators to cooperate in a shared configuration environment

- **Global service inventory:** TSB's management plane offers a comprehensive view of every application, wherever it runs––in modern or traditional deployments––including real-time health, endpoints, and performance. Whether in data centers or the cloud, VMs and containers are observable and managed with a consistent set of tools and processes.

- **Configuration safeguards:** TSB's configuration validation ensures correctness by construction. Service-level isolation and organizational controls guarantee that only correct configuration reaches production workloads.

- **Comprehensive Istio & Envoy lifecycle management:** Tetrate takes the complexity out of upgrading Istio and Envoy everywhere. Lifecycle is managed centrally and may be upgraded incrementally with a full inventory of mesh deployments, versions, plus current and auditable historic state.

Security

Tetrate simplifies policy enforcement by extending the capabilities of a single-cluster service mesh to an entire application fleet. App-level zoning allows for secure, fine-grained segmentation. Vetted workflows allow application, platform, and infosec teams to effectively manage policies for the entire organization. A centralized view of config changes with policy controls enables audit and continuous proof of compliance.

- **Compliance:** Tetrate Service Bridge works with the FIPS and federally certified Istio builds created and supported by Tetrate. It offers PCI DSS conformance, plus out-of-the box controls to ensure compliance with regulatory requirements. Audit log exports are also available to provide proof of current and historical adherence to governance and compliance standards.

- **End-to-end mTLS:** Tetrate Service Bridge integrates with existing public key infrastructure––including ACM and Venafi––for centralized management. And TSB supports Istio's ability to enable encryption to be implemented consistently and flexibly across all workloads, including between containers and VMs. Mutual TLS enforcement can be applied by security teams according to governance policy, without relying on the capacity and timelines of app teams to build support for it into each application.

- **Secure access:** Tetrate's built-in implementation of NIST's next-generation access control (NGAC) provides for fine-grained, flexible segmentation, authentication, and authorization. Move auth out of your applications to unburden your developers. Perform access control in Envoy between services to ensure consistent policy enforcement across your entire fleet and manage it all in one place.

- **Policy-driven:** The mesh takes cross-cutting security concerns out of the application code stack and into a dedicated infrastructure layer where they belong. Security teams have centralized control of policy for every application, regardless of which language or technology stack they use. This gives security teams the visibility and control they need to keep applications and data safe, while freeing up app dev cycles to return focus to creating business value.

Connectivity

While single-cluster service mesh facilitates connections between workloads, TSB enables seamless connectivity across heterogeneous, multi-site deployments. TSB offers simple, consistent workflows for managing connectivity, regardless of where applications are deployed and what compute they run on.

- **Multi-cluster, multi-platform, multi-cloud:** From the start, Tetrate's mission has been to connect hybrid networks and compute environments--including multiple clusters, multiple clouds, and on-premises--and to operationalize application networking in a simple, consistent way across an entire fleet. Now, Tetrate brings infrastructure-agnostic communication and operation of the stack.

- **Seamless communications and management:** Runtime service discovery for all apps allows for seamless application communication spanning an entire enterprise infrastructure, from multi-cluster to multi-cloud, monoliths to microservices. The global service inventory allows for mapping application components to the right teams with the right tools to ensure apps can be managed efficiently and safely.

- **Standardized:** Ship code faster by standardizing the way your applications communicate. Replace patchworks of custom gateways. Instead, offload security, connectivity, and observability to the mesh to increase agility and operational efficiency across your organization.

Observability

Tetrate's observability capabilities offer a comprehensive view of application metrics, allowing operators and app teams to understand dependencies and application health at a glance. This whole-app context makes it easier to troubleshoot and can reduce mean time to recovery. Tetrate's observability tooling facilitates applying service level objectives globally and consistently to all applications, without having to build instrumentation into the application stack.

- **Consistent SLOs:** Being able to see the topology of services and their dependency relationships makes it easier to measure and understand application health at a glance. Correlated metrics, traces, logs, and lifecycle events make it easier to troubleshoot apps and reduce MTTI and MTTR.

- **Consistent signals:** Service mesh provides a way to get a consistent set of both service and app-level signals from all services in a cluster. Tetrate offers that same consistency across all applications and ensures alerts get to the right teams, so they can take action quickly when issues arise and before downstream customers are affected.

- **Behaviors:** TSB allows operators to collect consistent baseline metrics for all apps without having to get them from each app team. They can create, measure, and monitor both app and service-level SLOs for every application and use that in-depth knowledge of app behaviors to recognize anomalies and take action before users notice.

Reliability

TSB provides built-in high-availability capabilities layered over Istio's reliability primitives. Its distributed, autoscaling ingress layer plus tools to manage transparent failover—from a manual "big red button" to touch-free, automatic failover—allow for traffic shifting to healthy resources. The context of running services provided by TSB, including their dependencies and where they're deployed, helps teams to understand the global health of their applications, measure how their apps behave, and equip them with the tools needed to build resilient systems.

- **Release cadence:** Built-in tools let you measure when it's safe to release newly deployed versions of your services to live traffic. Tetrate Service Bridge allows developers to move quickly, but safely roll back to a known-good state when needed.

- **Traffic migration:** App teams have the tools they need to understand their applications' health and take action on early indicators, with the best information possible at their fingertips.

- **Lifecycle:** TSB reduces the onerous labor and mitigates the risks of updating new app versions as well as the underlying mesh infrastructure. App teams get primitives for fast, safe application updates. Platform operators get tooling to keep application infrastructure up-to-date.

- **Load balancing:** Offload reliability concerns to the mesh by using built-in recovery features like timeouts, retries, and circuit breakers. Increase the ability to mitigate cascading failures with client-side load balancing. Decrease latency and minimize egress costs with locality-aware load balancing that ensures local traffic stays local.

06 Tetrate Istio Subscription

Tetrate Istio Subscription (TIS) has everything you need to run Istio and Envoy in highly regulated and mission-critical environments. It includes Tetrate Istio Disto, a 100% upstream distribution of Istio and Envoy that is FIPS-verified and FedRAMP ready and provides vetted builds of Istio tested against all major cloud platforms. For teams requiring open source Istio and Envoy without proprietary vendor dependencies, Tetrate offers the only 100% upstream Istio enterprise support offering.

TIS is fully compliant with NIST 207A, the publication that defines the U.S. federal standard for the zero trust architecture that all federal applications must begin following. In fact, Tetrate worked directly with NIST in developing the standard. Additionally, TIS is designed for highly regulated industries where security is paramount. It allows organizations with globally distributed cloud infrastructure and a large, diverse group of application development organizations to automate policies that assure compliant security – even if attackers manage to get inside their networks.

TIS provides organizations with additional features and enterprise-grade services for their Istio deployments. TIS provides extended Istio version support and CVE fixes beyond upstream Istio (release date plus 14 months). The Istio distributions are hardened and performant, and are full distributions of the upstream Istio project.

Key features of the Tetrate Istio Subscription include:

- **Hardened, performant distribution of 100% upstream Istio and Envoy, suitable for FedRAMP:** Upstream Istio doesn't provide FIPS-compliant builds suitable for use in regulatory environments. Tetrate solves this problem by offering TID Istio and Envoy binaries that are compiled against FIPS-validated cryptographic modules and verified by an accredited testing laboratory to be compliant with programs like FedRAMP. TIS subscribers get access to Tetrate's FIPS-verified Istio builds and the corresponding certification of compliance.

- **Compatibility testing for the major cloud providers including AWS, Azure and GCP:** To ensure a seamless experience in the environments you actually use, Tetrate Istio Distro is extensively tested for performance and compatibility across multiple Kubernetes versions on AWS, Azure, and GCP.

- **Improved Installation and lifecycle management:** Tetrate's open source management tooling makes it easy to install, manage, and upgrade Istio. Tetrate Istio Distro comes with the getmesh CLI to quickly and reliably maintain installations, apply upgrades, and perform quick troubleshooting operations. For Amazon EKS and EKS Anywhere users, Tetrate offers one-click Istio installation and support from the EKS Marketplace.

- **Enhanced Security:** TIS includes additional security features and best practices to help organizations secure their microservices communications effectively.

- **Performance and Scalability:** Tetrate provides performance enhancements and scalability guidance to optimize Istio's performance for large-scale applications.

- **Regular Updates:** Receive access to regular updates, bug fixes and feature enhancements to keep Istio deployments up-to-date.

- **Expanded version support and maintenance:** TID Istio binaries have a longer support window than Istio community builds to reduce operational overhead. Tetrate supports and maintains the four most recent versions of Istio–twice that of the Istio community project–which translates to an extra six months of support and maintenance. TID significantly reduces the burden to stay up-to-date on community-supported versions, allowing you to manage updates on a predictable schedule.
- **Enterprise-grade Support:** Professional support and assistance with your Istio deployment such as troubleshooting, issue resolution and guidance on best practices.
- **Access to Expertise:** Gain access to Tetrate's team of service mesh experts who provide guidance and consulting on implementing Istio effectively.

Users of Tetrate Istio Subscription benefit from:

- **Tetrate Istio distributions:** whenever you need an Istio distribution that is tested for use in AWS, Azure, GCP or vanilla Kubernetes
- **FIPS-validation:** FIPS-validated builds meet the needs of FedRAMP and other compliance-sensitive environments
- **Supported Add-Ons and Integrations:** for easy deployment, integration and extensions
- **Support from Tetrate Experts:** our expert technical support team are experienced Istio users and contributors; it's like having an Istio expert embedded within your own team!

07 Service Mesh Adoption

However you choose to tackle service mesh adoption, there is a fundamental set of concerns that need to be addressed to be successful:

- **Cultural change:** Introducing a new infrastructure component, especially something as fundamental as a service mesh, always requires cultural change. One of the core powers of the service mesh is that it centralizes control and enables small, specialized teams to impact the entire organization. It may require a shift in an organization's engineering culture to fully realize the benefits of this capability.

- **Education:** As we introduce new infrastructure with new capabilities, we need to educate different stakeholders within the organization on those capabilities and how they can leverage them. In the case of the service mesh, which has far-reaching capabilities, this means working with the platform, security, networking, application developers, and operations teams – nearly the entire IT organization.

- **Operationalization:** Any new infrastructure component requires some degree of time, effort, and experience to mature from a proof of concept to a fully operational part of critical production infrastructure. In addition to training and education, new infrastructure requires additional processes and tooling to be developed, including playbooks, dashboards, and alerting. Provisions must be made for edge cases that don't arise in test environments. Over time, we build expertise and increasing confidence in operating the system.

  Another key element of operationalizing the service mesh is to abstract away its moving parts from application developers. This gives more consistency and homogeneity, which makes adoption simpler.

- **Tooling:** A key component of operationalization is developing (or augmenting existing) tooling, processes, and pipelines. The platform and operations teams will need tools to understand and troubleshoot production issues. Application developers will need tools to understand how their application interacts with the mesh, sometimes in unexpected ways. There will likely be changes to CI/CD pipelines and so on.

  Operationalizing also means leveraging net-new capabilities the service mesh provides. For example, consistent metrics for all applications enables you to offer a single, consistent, reusable set of dashboards to production and application teams, reducing the time it takes to bring new services into production and improving their overall production operability once they get there. The same is true for logging, tracing, security controls, and capabilities like safe deployments via canaries.

- **Process change:** As with any infrastructure investment, these education, operationalization, and tooling updates, and the cultural adaptations that accompany them, will require some process changes to fully realize the value of a service mesh.

We see organizations adopt the mesh most successfully when they minimize changes to existing processes for applications development teams. As your investment begins to demonstrate value to the organization in concrete ways—e.g., enabling encryption in transit for all applications without needing to involve application developers—you develop the political will within your organization to introduce larger process changes, which will, in turn, unlock more value from your existing infrastructure investment, in a virtuous cycle.

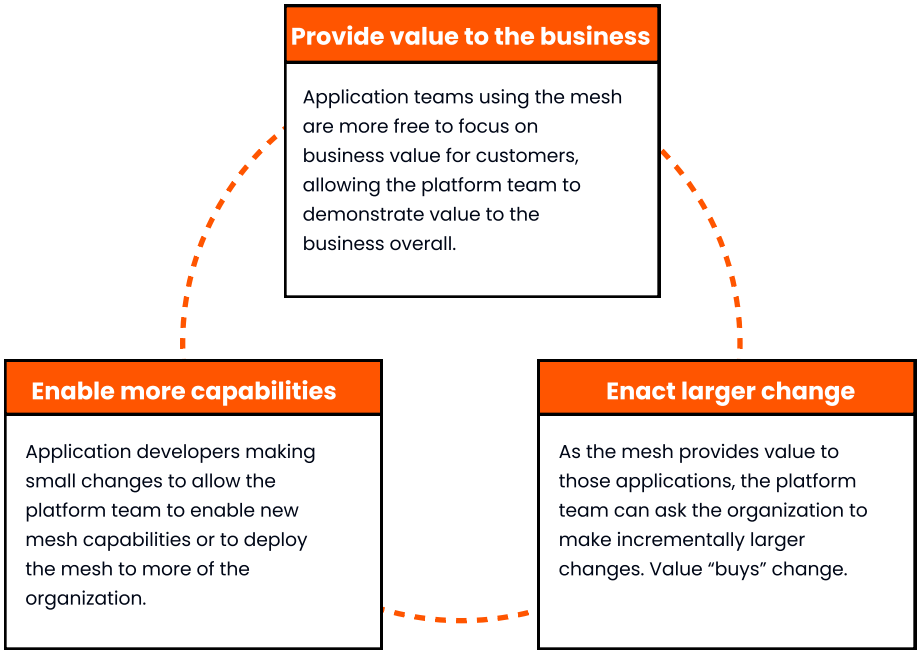


Fig 1: The virtuous cycle of providing value to the business, which provides the political capital to ask for slightly larger changes for the sake of the platform, which allows the platform team to enable new or more capabilities, which then provides more value to the business.

- **Time to value:** Perhaps the most important concern for any infrastructure improvement is the time it takes to demonstrate value. Showing a steady stream of incremental value early on builds the political currency to "pay for" larger process and cultural changes to the organization which, in turn, allow you to show greater value. Those increments may often have a large impact.

Platform-Driven Adoption

To understand the typical mesh adoption pattern, it's instructive to cover the general adoption flow we see used by most of our customers. The service mesh adoption effort tends to be driven by a platform team whose purview includes how application developers get their apps to work in the organization. The platform team works closely with other central teams in the organization to prepare for service mesh adoption.

Team	Typical Collaborations
Security	• The security model for the platform, including the mesh • Integrations with existing public key infrastructure (PKI) • Overarching policies like how the mesh works across segmented networking architectures with a DMZ, application, and data zones

Networking	• Establish the pattern for integrating the mesh with existing networking infrastructure, including:
	  • Load-balancing appliances
	  • Shared API gateways
	  • WAFs
	  • GSLB
	  • Existing TLS termination points

Infrastructure	• Capacity planning
Operations	• Training programs • Runbook development
Application	• Preparation of early-adopter application teams

After aligning these stakeholders, the key decision point becomes: how do you onboard applications into the mesh? Getting applications onto the mesh, and getting application developers to leverage the capabilities they get as a result, is the long tail of any mesh adoption. However, as the service mesh platform gains more adoption within the organization, it will radically streamline the cross-team collaboration required to deploy and manage applications.

Where application teams were previously required to interface with all of the central teams individually, the service mesh platform and the team that runs it become a central hub to facilitate the concerns of security, networking, operations, and infrastructure teams, while application teams need only interact with the platform. This greatly simplifies application teams' responsibilities, reducing operational overhead while increasing agility.

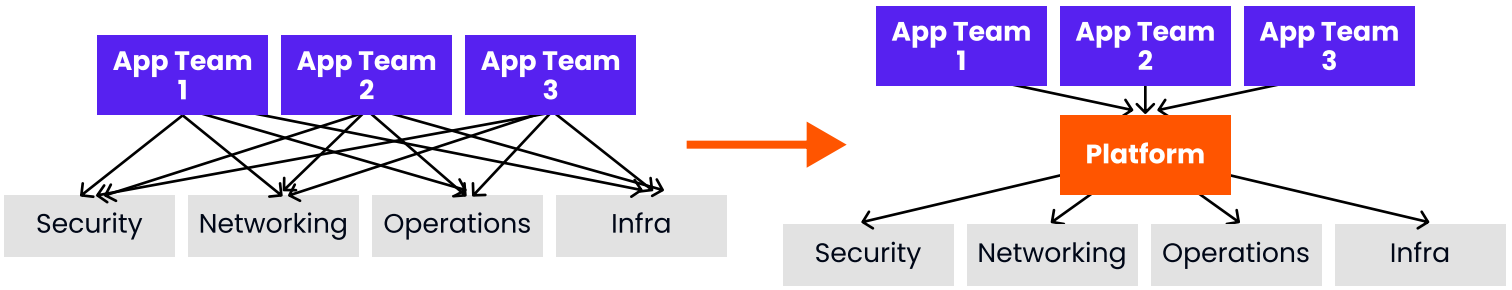


Figure 2: The platform streamlines cross-team collaboration

A Phased Approach: Adopting the Mesh, Application by Application

Based on years of experience helping organizations build a service mesh practice, we recommend a phased approach whereby we bucket applications into groups sharing certain common attributes, then onboard one application group at a time. This allows us to realize many benefits of the mesh and establish general practices for that entire group of applications. We also build production and operational experience that we can apply to enhance the onboarding experience of the next group. Each new group of applications we onboard tends to be incrementally more critical, or presents additional complexities compared to the group before.

We commonly bucket applications into groups along the following vectors:

- Application criticality
- Runtime requirements, including:
  - PCI vs non-PCI
  - Network zone: DMZ vs application vs data tier
  - Deployment environment: containers vs Kubernetes vs VMs vs bare metal
  - Compute architecture: ARM vs x86
  - Low-latency/high-performance vs. less demanding
- Language and framework, e.g. J2EE, Spring Boot, .NET, Node, golang, etc.
- Protocol:
  - Layer 7/request-oriented protocols like HTTP, gRPC, websockets, Thrift, etc.
  - Layer 4/stream-oriented protocols like MySQL, Redis, general TCP, etc.
- Other concerns: organizational (e.g. grouping by business unit), disaster recovery strategy (failover-based, active-active deployment, etc).

This exercise tends to yield between 10-20 groups of applications. We then start with less critical, "well-behaved" applications like newly minted microservices in Kubernetes, then move on to services in common frameworks that speak HTTP/REST, and eventually VM and bare-metal applications. We work our way up incrementally until we reach the most critical applications with the most stringent runtime requirements.

Anecdotally, we see the 80-20 rule at play frequently here, where roughly 80% of applications can be onboarded with relative ease, while the remaining 20% require additional tooling and effort and, in some cases, changes to the application itself.

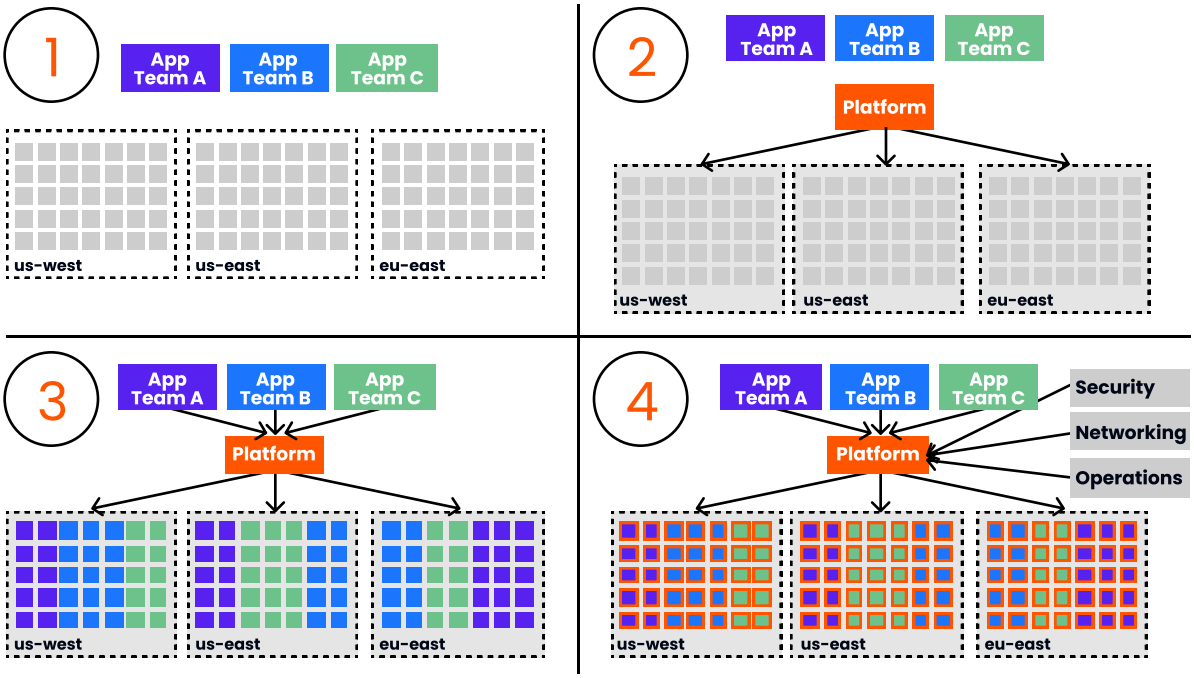


Figure 3: An effective platform helps hide infrastructure concerns from application teams, provides tenancy and ownership for applications running on that infrastructure, and acts as a single integration point for central teams to enforce policy across all applications, without needing to involve application teams (most of the time).

Benefits of incremental adoption along the axes of concern

- **Cultural change:** The biggest virtue of a phased approach is the limited organizational and cultural change needed in the early stages, before the mesh has clearly demonstrated its value. Only a select few application teams at a time need to be educated and enculturated, making the process more manageable for organizations with many teams.
- **Education:** As with cultural change, the educational lift is bounded: only a select few teams at a time need to be educated, and typically the platform team can control which teams those are.
- **Operationalization:** The biggest benefit of the phased approach is gaining experience, skills, and confidence with the new system in a low-risk environment first, then gradually increasing the risk by onboarding new, more critical application groups only as confidence in operations

and operational expertise increases. Additionally, you can realize many benefits of the mesh for a group of applications—eg., mTLS, and metrics, and logs, and traffic controls—rather than only one application at a time.

- **Tooling:** Onboarding similar applications together creates an opportunity to develop tooling, processes, and templates that work effectively with that group of applications. Building this "happy path" reduces friction for application teams and accelerates their ability to realize value from the mesh.
- **Process change:** Working with smaller groups of teams and specific types of applications gives us more leeway to ask them for process change. We'll need that leeway, because the first few groups of applications are where the platform team becomes familiar with the mesh and develops its operational muscle. This will be especially apparent in the first few groups, and as we build out the tooling and processes for that group that will become common across the organization. As the platform team builds a practice of onboarding new applications, doing so becomes easier, with fewer process changes required.
- **Time to value:** Perhaps the most important result of this approach is generating a steady stream of value for the organization throughout the entire mesh adoption process. Onboard "easy" applications quickly and realize value right away as you develop the tooling and processes needed to bring more challenging groups of applications into the mesh.

Challenges of incremental adoption

- The total amount of time to adopt a mesh across the entire organization can be longer than a "big bang" approach. However, you'll show initial value more quickly, and benefit from a continuous stream of value at more regular intervals.
- Covering the last 20% of applications may be a long process taking roughly as much work as the entire first 80%.

Special Case: a "Big Bang" for Homogenous Application Environments

While not the most common case, a significant number of organizations we work with have relatively homogenous application stacks. For example, most of their business applications might use Spring Boot and run in Cloud Foundry. In the same vein, some organizations have invested in strong platform abstractions already. In these cases, we can take a slightly different approach to service mesh adoption, which we call the "big bang". We do more prep work up front, but that work enables us to onboard a large swathe of the organization onto the mesh in a single push.

As a result, it takes less time overall to complete rolling out the service mesh to the entire organization, at the expense of greater effort up front and a longer time to realize initial value. In this case, we typically focus on deploying one or two key capabilities or features of the mesh for all applications at once, rather than deploying many features of the mesh for a single group of applications before we move on to the next group, as in the phased approach. For homogeneous application environments, this is equivalent to executing the phased approach where the organization only has two to three buckets of applications.

Challenges of the Big Bang approach

- Some parts of the mesh adoption cannot be done centrally: for example, implementing and maintaining access control policies for every service. At some point, even with a wide approach, you have to bring application teams into the fold with training and guidance, as well as the ability to (safely) change that configuration.
- There is more initial stress on the platform and operations teams, as they have much more exposure to the entire organization but have had less time to adopt new operational practices.

How to Choose an Adoption Pattern

For organizations with a wide variety of applications, many application teams, and highly critical applications, we recommend the phased approach as the safest and surest path to successful adoption and operational readiness of the service mesh. It is the default path we recommend to enterprises looking to adopt a service mesh.

For organizations with strong existing platform abstractions, or those that have a homogenous application mix, the “big bang” approach requires less overall work than an incremental approach and should be seriously considered. However, you cannot escape some amount of incremental work for individual application teams to fully realize the value of the service mesh investment.

08 Appendix

The Essential Service Mesh Adoption Checklist

Here we've included a high-level checklist of the typical execution path for service mesh adoption. While not exhaustive, it should serve as a useful set of guideposts as you're developing your own service mesh adoption plan.

Preparing your teams

Prepare the teams you'll be working with, usually concurrently. The goals for this stage are to establish how the service mesh will fit into your existing infrastructure and processes. You'll know you're ready to move forward when you've established a plan for each of the key areas we list below, and the teams you're working with are committed—and, ideally, excited—to adopt the mesh and experience its benefits for themselves.

Security

Work with the security groups and the CISO office to:

Build the mesh security model for your organization. A few important areas to explore:

- [ ] How does the mesh fit in with the DMZ?
- [ ] How does the mesh fit with the existing firewall/WAF?
- [ ] How does the mesh fit with existing SSL termination?
- [ ] How does the mesh fit with existing API Gateway appliances?

Develop a PKI/certificate strategy for the mesh. Determine what integration work is needed:

- [ ] Will the mesh issue certificates out of the organization's existing root, or from a separate root of trust?
- [ ] Will the mesh issue certificates or will you use existing certificate infrastructure for workload certificates? Depending on the approach, this may require changes to those certificates for the service mesh's authentication capabilities to work.
- [ ] How will we choose to deliver signing certificates to the mesh control plane?

Determine what policies the mesh will enforce from the outset.

Onboard service mesh containers into your system, with all required security scanning.

- [ ] Put a process in place to keep these images up-to-date at a regular cadence. For reference, Istio releases four times a year.

- [ ] Put a process in place to address CVE/image scanning and pentest results for the mesh infrastructure.

Networking

Work with the networking team to:

Assess how the mesh changes networking models in your organization:

- [ ] For example, you may eliminate hairpin traffic from apps that communicate "internally" by going out to a global load balancing system and back through the DMZ and into the datacenter, even when the two workloads run on the same rack. This will have an impact on networking and security.

- [ ] Determine where TSL (SSL) termination needs to happen. In existing load balancers, offload it to the mesh, or even the application.

- [ ] Determine how you will manage and deploy the mesh's gateways.

- [ ] Determine your gateway deployment model: one gateway per team; a single shared gateway for all teams; or mostly shared, with a few standalone. We recommend a gateway per team to start with.

Develop a strategy for how to integrate the mesh with other networking components.
For example, how does the mesh fit with existing API Gateway appliances?

Capacity Planning

Work with your capacity planning teams to ensure you're ready for the service mesh.

- [ ] The overhead of the service mesh can vary, but we typically see ~10% overhead, with some applications seeing as much as 30% depending on use cases and workloads. In our experience, existing systems tend to be overprovisioned to the extent that the mesh infrastructure can be added without meaningful changes to resource planning in the short to medium term.

Operations

Work with the operations team to prepare them for incoming infrastructure changes.

- [ ] Start to familiarize them with the service mesh and the capabilities it brings if they were not part of the buying decision.
- [ ] Begin education and training with them, the earlier the better.

Observability and Monitoring

Work with your observability and monitoring teams to prepare them for incoming infrastructure changes.

- [ ] Determine how you'll integrate the mesh's metrics with the existing infrastructure.
- [ ] Ensure you have capacity for the additional metrics the mesh will produce.
- [ ] Evaluate log levels and the resulting log volume produced by the mesh. Verbose logging in the mesh can produce a lot of data!

Platform

Prepare your larger platform team for the incoming infrastructure changes.

- [ ] Get every team member hands-on with your chosen mesh solution.
- [ ] Begin education and training for the platform team ASAP.
- [ ] As a goal, we recommend that a handful of platform team members engage in the underlying OSS projects. This will increase organizational familiarity, confidence, and ability to operate with those technologies.
- [ ] Develop the tenancy model for your system:
  - [ ] Do teams own namespaces? Whole clusters?
  - [ ] How does that map to the mesh?
  - [ ] Will you share ingress gateways or give an ingress gateway per team? We recommend a dedicated ingress gateway per team to start with.

Initial Onboarding

Once your teams are prepared and ready to start implementation, it's time to start the initial phase of adoption.

Bucket your applications by criticality, runtime requirements, frameworks and protocols.

- [ ] Determine the order in which you'll attack the buckets. We recommend starting with simpler applications that have low criticality and more relaxed runtime requirements—e.g., non-PCI or with loose latency requirements.
    - In the organizations we work with, this group often includes the first Kubernetes services being developed.
    - If your first bucket contains a large group of applications, we recommend hand-selecting an initial subset—between three to five teams—to work with closely and directly.

Begin to roll out the mesh one bucket at a time:

- [ ] Work closely with your initial teams to develop tools, practices, process, and expertise for this class of applications.
    - [ ] Ingress gateway strategy: shared or dedicated to the team?
        - Decide based on criticality and production experience; favor dedicated gateways until you're very confident.
        - Highly critical applications should almost always get a dedicated gateway.
    - [ ] CI/CD changes.
        - If the application team will be authoring mesh configuration, that configuration should be deployed by the same continuous deployment system that rolls out their application.
    - [ ] Understand any application changes that may be required:
        - See Istio's documented pod requirements.
        - Do you need to change the application's port labels, or turn off TLS in the application to allow the mesh to do it?
        - How do we deploy the sidecar itself? Do they opt in? Opt out?
        - For this set of apps, are there any systemic changes that need to be made—e.g., exempt database traffic for the time being?
- [ ] Deploy dashboards for the application based on mesh metrics.

Execute the mesh onboarding. This will usually take a few deployment cycles, as you want to gather information and expertise before, for example, requiring mTLS everywhere. The major steps for each application are:

- [ ] Start with PERMISSIVE mTLS for the application.
    - This will ensure non-mesh applications calling this application do not break.
- [ ] Configure gateway routing.
    - If you decide to use dedicated gateways, deploy the gateway instances themselves.
- [ ] Add sidecar proxies to the application instances (enable injection).
- [ ] Add a DestinationRule requiring mesh mTLS.
    - This ensures applications in the mesh are using mTLS even if the server allows non-mTLS clients as well.
- [ ] Configure traffic management settings.
- [ ] Configure rate limiting and authentication policies.
- [ ] Author authorization policies restricting which clients can communicate.
    - These are typically written by the app team.
- [ ] Lock down STRICT mTLS as soon as all clients have upgraded.

Then apply those learnings to the entire bucket of apps.

- [ ] Create a rollout template that encodes the happy path for all applications in that bucket.
- [ ] Create playbooks for executing the onboarding.

Repeat with the next bucket of more critical, higher runtime requirement, “harder” applications, using the previous learnings to make it easier and faster to execute.

Tetrate Academy

If you are new to service mesh and Kubernetes security, we offer free online courses at Tetrate Academy that will quickly get you up and running with Istio and Envoy. Our courses are expertly curated, hands-on training experiences from the co-creators of open source Istio and Envoy. Private training for enterprise customers available upon request. https://academy.tetrate.io

Get Started with TIS

If you're looking for a fast way to get to production with Istio, check out Tetrate Istio Distribution (TID), Tetrate's hardened, fully upstream Istio distribution, with FIPS-verified builds and support available. It's a great way to get started with Istio knowing you have a trusted distribution to begin with, an expert team supporting you, and also have the option to get to FIPS compliance quickly if you need to.

Get Started with TSB

As you add more apps to the mesh, you'll need a unified way to manage those deployments and to coordinate the mandates of the different teams involved. That's where Tetrate Service Bridge comes in. Learn more about how Tetrate Service Bridge makes service mesh more secure, manageable, and resilient here, or contact us for a quick demo

Additional Resources

Please visit https://tetrate.io/resources/